Invernadero IoT System

Este proyecto implementa un sistema de Internet de las Cosas (IoT) para un invernadero. El sistema recoge datos de temperatura de los sensores, los procesa y notifica a los suscriptores sobre los eventos relevantes.

Clases y Funciones

El sistema se compone de varias clases y funciones:

Sensor

La clase `Sensor` simula un sensor de temperatura en el invernadero. Cada sensor tiene un ID y una lista de historial de temperaturas. Los sensores envían información de temperatura al servidor de Kafka cada 5 segundos.

Suscriptor y DueñoInvernadero

La clase `Suscriptor` es una interfaz para los suscriptores que desean recibir notificaciones del sistema. La clase `DueñoInvernadero` es una implementación concreta de un suscriptor.

Handlers

Las clases `Handler` y sus subclases implementan el patrón de diseño Chain of Responsibility. Cada handler procesa los datos del sensor de una manera específica y pasa los datos al siguiente handler en la cadena si no puede procesarlos.

- `StrategyHandler` es una interfaz para los handlers que implementan diferentes estrategias de procesamiento de datos.
- `StrategyMeanStdev`, `StrategyQuantiles` y `StrategyMaxMin` son implementaciones concretas de `StrategyHandler` que calculan diferentes estadísticas de los datos del sensor:
- `TemperatureThresholdHandler` y `TemperatureIncreaseHandler` son handlers que verifican si la temperatura actual supera un umbral específico o si la temperatura ha aumentado significativamente en un corto período de tiempo, respectivamente.

IoTSystem

La clase `IoTSystem` es la clase principal del sistema. Implementa el patrón de diseño Singleton para asegurar que sólo haya una instancia del sistema. El sistema tiene una lista de suscriptores y una lista de handlers para procesar los datos del sensor. Los suscriptores pueden ser añadidos y eliminados del sistema. Cuando el sistema recibe nuevos datos del sensor, los procesa con los handlers y notifica a los suscriptores.

Ejecución

El script principal crea una instancia del sistema, añade un suscriptor, procesa algunos datos del sensor y notifica al suscriptor. Luego, elimina al suscriptor y trata de notificar de nuevo, lo que no debería tener ningún efecto ya que no hay suscriptores.

Dependencias

Este proyecto utiliza las siguientes bibliotecas de Python:

- **`datetime`** y **`time`** para trabajar con fechas y tiempos.
- **`random`** para generar datos de temperatura aleatorios.
- **`statistics`** para calcular estadísticas de los datos del sensor.
- **`abc`** para definir clases abstractas.
- **`confluent_kafka`** para enviar y recibir datos a través de Kafka.
- **`json`** para serializar y deserializar datos en formato JSON.
- **`pandas`** para trabajar con datos en formato de serie y calcular cuantiles.
- **`pytest`** para pruebas unitarias.

Repositorio utilizado para realizar el proyecto

Repositorio de Github